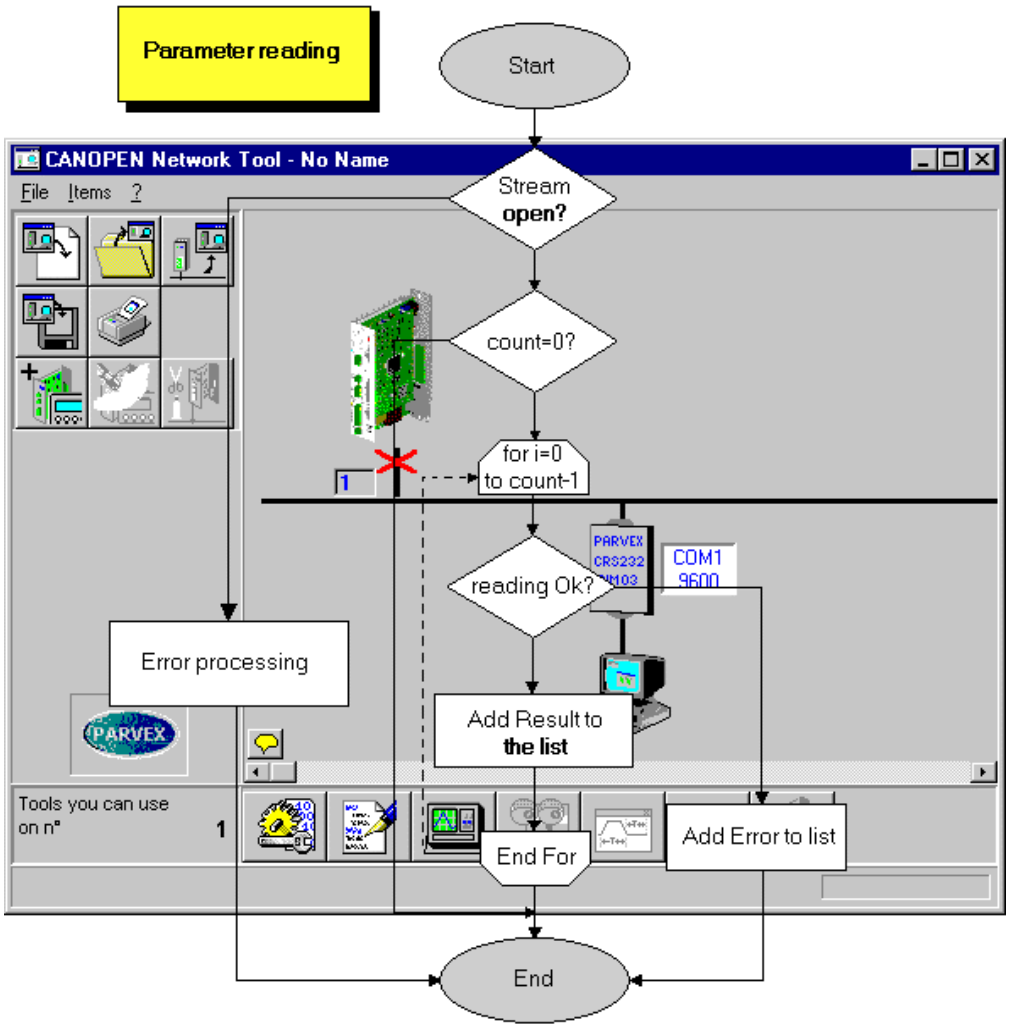
SSD Parvex SAS
8, avenue du Lac - B.P. 249
F-21007 Dijon Cedex
www.SSDdrives.com

DIGIVEX MOTION

PME Tool Kit

User and commissioning manual

PVD 3528 GB – 11/2003



PRODUCT RANGE

1 - « BRUSHLESS » SERVODRIVES

TORQUE OR POWER RANGES

- **BRUSHLESS SERVOMOTORS**, LOW INERTIA, WITH RESOLVER

  Very high torque/inertia ratio (high dynamic performance machinery):

⇒ NX -HX - HXA — 1 to 320 N.m

⇒ NX - LX — 0,45 to 64 N.m

  High rotor inertia for better inertia load matching:

⇒ HS - LS — 3,3 to 31 N.m

  Varied geometrical choice :

⇒ short motors range HS - LS — 3,3 to 31 N.m

⇒ or small diameter motors : HD, LD — 9 to 100 N.m

  Voltages to suit different mains supplies :

⇒ 230V three-phase for «série L - NX»

⇒ 400V, 460V three-phase for «série H - NX»

- **"DIGIVEX DRIVE" DIGITAL SERVOAMPLIFIERS**

⇒ SINGLE-AXIS DSD

⇒ COMPACT SINGLE-AXIS DµD, DLD

⇒ POWER SINGLE-AXIS DPD

⇒ MULTIPLE-AXIS DMD

- "PARVEX MOTION EXPLORER" ADJUSTING SOFTWARE

2 - SPINDLE DRIVES

- **SPINDLE SYNCHRONOUS MOTORS**

⇒ "HV" COMPACT SERIES

⇒ "HW" ELECTROSPINDLE,frameless, water-cooled motor — From 5 to 110 kW up to 60,000 rpm

- **"DIGIVEX" DIGITAL SERVOAMPLIFIERS**

3 - DC SERVODRIVES

- **"AXEM", "RS" SERIES SERVOMOTORS** — 0.08 to 13 N.m
- **"RTS" SERVOAMPLIFIERS**
- **"RTE" SERVOAMPLIFIERS** for DC motors + resolver giving position measurement

4 - SPECIAL ADAPTATION SERVODRIVES

- **"EX" SERVOMOTORS** for explosive atmosphere
- "**AXL" COMPACT SERIES SERVOREDUCERS** — 5 to 700 N.m

5 - POSITIONING SYSTEMS

- **Numerical Controls** « **CYBER 4000** » 1 to 4 axes
- **"CYBER 2000"** NC 1 to 2 axes
- **VARIABLE SPEED DRIVE - POSITIONER**

⇒ SINGLE-AXIS DSM

⇒ POWER SINGLE-AXIS DPM

⇒ MULTIPLE-AXIS DMM

- ADJUSTMENT AND PROGRAMMING SOFTWARE PARVEX MOTION EXPLORER

CONTENTS

1. GENERAL POINTS 2

1.1 List of published DIGIVEX MOTION manuals 2
1.2 CANopen protocol 2
1.3 Presentation 3
1.4 General principles 4

2. USER'S GUIDE 5

2.1 Required files 5
2.2 Interface D.L.L. PXCOMM.DLL 6
2.3 PC-CRS232 software driver 6
2.4 Parameter definition file structure 7
2.5 Memories 8
2.5.1 EEPROM_DM 8
2.5.2 RAM_DM 8
2.5.3 FLASH_DM 8
2.5.4 PROG_DM 8
2.5.5 FIRM_DM 8
2.5.6 RAM_PC 8
2.5.7 Bulk Storage 8

3. GUIDE TO FUNCTIONS 11

3.1 Alphabetical function directory 11
3.2 Function feedback - error codes 12
3.3 Detailed description of the various functions 12
PxDirectWriteParam 18
PxOpenStreamEx 22

4. PROGRAMMING EXAMPLES 32

Characteristics and dimensions subject to change without notice

YOUR LOCAL CORRESPONDENT

SSD Parvex SAS
8 Avenue du Lac / B.P 249 / F-21007 Dijon Cedex
Tél. : +33 (0)3 80 42 41 40 / Fax : +33 (0)3 80 42 41 23
www.SSDdrives.com

1. GENERAL POINTS

1.1 List of published DIGIVEX MOTION manuals

♦ DIGIVEX Single Motion (DSM) User Manual	(DSM)	PVD3515
♦ DIGIVEX Power Motion (DPM) User Manual	(DPM)	PVD3522
♦ DIGIVEX Multi Motion (DMM) User Manual	(DMM)	PVD3523
♦ DIGIVEX Motion - CANopen		PVD3518
♦ DIGIVEX Motion - Profibus		PVD3554
♦ PME-DIGIVEX Motion Adjustment Manual		PVD3516
♦ DIGIVEX Motion Directory of Variables		PVD3527
♦ DIGIVEX Motion Programming		PVD3517
♦ DIGIVEX Motion - Cam Function		PVD3538
♦ PME Tool kit User and Commissioning Manual		PVD3528
♦ CANopen - CAN Bus Access via CIM03		PVD3533
♦ CANopen - Remote control using PDO messages		PVD3543
♦ "Block Positioning" Application Software		PVD3519
♦ "Fly shear linear cutting" software application		PVD3531
♦ "Rotary blade cutting" software application		PVD3532

1.2 CANopen protocol

The CANopen protocol is used for parameter setting and supervision of the Digivex Motion family of devices.

The CANopen protocol allows users to control, among other things:
- process messages (PDO),
- service messages (SDO).

PDO type messages are not cleared whereas SDO type messages are.

SDO type messages are used to control ad hoc instructions sent to a single addressee (a variable or a parameter of a subscriber in the network is read or written). At maximum CAN bus baud rate (1 Mbit per second) it is theoretically possible to send a message every millisecond. When using software under Windows to control these messages, this potential is in practice limited to sending one message every 50 ms.

PDO type messages are used for sending high baud rate cyclical instructions (see PVD3543 "Digivex Motion – Position Control with Interpolation – Using PDO messages")

1.3 Presentation

This document presents the **PME Tool Kit** and describes the method for accessing the CANopen network control functions defined by Parvex.

- **Section 1** presents the *General Points* about **PME Tool Kit**.
- **Section 2** is the *User's Guide*. It shows the functional implementation of **PME Tool Kit**.
- **Section 3** is the *Guide to Functions*.
- **Section 4** presents some *Programming Examples*.

PME Tool Kit provides access from a PC with an RS232 serial link to the variables and parameters in a Digivex Motion positioner drive via a CRS232 interface module.

PME Tool Kit is a "DLL" type function library provided on the PME software installation CD-Rom (tool_kit directory). This library contains all the functions required for initialising the CRS232 module and for controlling SDO type messages carried over the CANopen bus.

PME Tool Kit is intended for users proficient in programming in a Windows® 32-bit environment (C, Delphi®, Visual Basic®, etc.). However, no knowledge of the CANopen protocol is required.

PME Tool Kit contains:

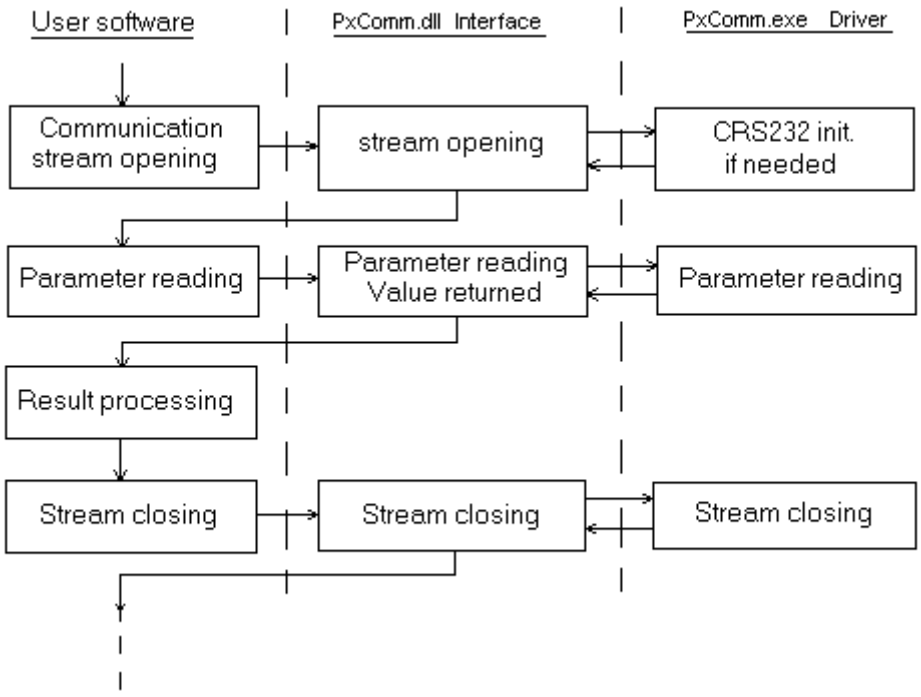
- A library containing all the CANopen network control functions in the form of a 32-bit **D.L.L.** (Dynamic Link Library),
- a CR232 interface **driver** in the form of a 32-bit executable,
- function, variable and parameter definition files,
- descriptions of all CANopen network control functions in documentary form,
- programming examples in several languages.

1.4 General principles

A **Stream** is a communication channel opened for a given CANopen subscriber.

The user must first of all open a stream. This stream is identified by a unique number assigned by the system. Once the stream is open, it is possible to read or write to or from a device by specifying its CANopen subscriber number and the stream number.

Any stream that has been opened must be closed to release the resources used. Sixteen communication streams can be opened at the same time in a Windows work session.



- When the stream is opened, a code identifying the variable and parameter definition file for the type of device concerned is sent to the DLL. It is in this file that the DLL finds a match for the variable or parameter number to be read or written and the associated information (CANopen index and sub-index numbers, format, minimum and maximum values, default value, etc.).

- Only one operation at a time can be done on a given stream. Unless otherwise specified, calling up DLL functions freezes the system, i.e. the program only resumes execution when the function is returned. A time-out mechanism allows the DLL to return an error if a function is not completed within a reasonable time on the network (subscriber absent, transmission error, etc.). This time-out varies with the function and the type of error encountered.

2. USER'S GUIDE

2.1 Required files

PME Tool Kit comes with the following files.

Caution! These files are designed for a Win32 development and working environment.

PxFnDef.pas	File representing the DELPHI unit containing the definitions of functions exported by the D.L.L. This file is used for programming in DELPHI language only. It is to be copied into the directory of the project being developed and must be placed in the **uses** clause.
PxFnDef.h	Pointer type definition file on the functions exported by the D.L.L. This file is used for programming in "C" language only. It is to be included in each source file of the "C" project using one of the DLL functions (**#include <PxFnDef.h>)**.
Apigid32.dll	Library file containing specific VISUAL BASIC conversion functions. This file is only used for VISUAL BASIC language programming. It must be copied to the development project directory.
PxComm.dll	Library file containing all the functions for CANopen network control defined by Parvex. This file must be placed in the same directory as the executables that use it.
PxComm.exe	Software drive for communication with a CRS232 interface via the PC serial link. This file must be placed in the directory containing PxComm.dll
Param.csv **Dsm.csv** **Dtp.csv**	These three files contain the definition of variables and parameters used by Parvex devices of the Digivex Motion family (number, index, sub-index, type, minimum value, maximum value, default value, etc.). Information provided by the CANopen protocol is controlled, so the presence of non-Parvex devices on the network can be detected. These files must be placed in the directory containing PxComm.dll
Firmcrs.h86 **Routram.h86**	These two files contain the executable code of the CRS232 interface. This code is loaded into the CRS232 interface when a communication stream is opened after the interface is first energised. These files must be placed in the directory containing PxComm.dll

2.2 Interface D.L.L. PXCOMM.DLL

This function library is written in DELPHI 4.0 for a Win32 environment.

All the functions are declared with the **stdcall** directive, making them directly accessible from other languages like "C".

The library must be integrated into the project under development. To do this, the function definition file **PxFnDef** must be included:

- either in a **uses PxFnDef** clause in DELPHI (**PxFnDef.pas** file)
- or by **#include <PxFnDef.h>** in C (**PxFnDef.h** file)
- or by copying the definitions contained in the Decla_VB_Function_Parvex.txt VISUAL BASIC file

The library must be in the same directory as the executable(s) calling its functions.

To access one of the variables or one of the parameters of a device, a decimal recording number corresponding to that variable or that parameter is passed on to a D.L.L. function (see PVD 3527, Digivex Motion Directory of Variables, column "no." in the descriptive tables). The communication driver must know the CANopen index and sub-index numbers of the variable or parameter to be able to access them. The library therefore matches these numbers via a variable and parameter definition file which must be placed in the same directory as the D.L.L.

PME Tool Kit comes with the variable and parameter definition files for Parvex devices in the Digivex Motion family (Param.csv, DSM.csv, DTP.csv). Some parameters defined by the CANopen standard are included in these files (type of device, manufacturer's name, etc.), which means the presence on the network of any device integrating these parameters can normally be detected.

2.3 PC-CRS232 software driver

The executable file **PxComm.exe** is the software driver that handles all communications between an application (via the interface D.L.L.) and an appliance on the CANopen network (via the CRS232 hardware interface).

This 32-bit executable should under no circumstances be closed while using Parvex CANopen network control functions. In normal use this program must be "iconised" and so invisible to the user.

The driver starts up automatically when an instance of the interface D.L.L. PxComm.dll is loaded into an application using at least one of its functions. It is closed when the last instance of the D.L.L. closes.

2.4 Parameter definition file structure

The Param.csv, DSM.csv and DTP.csv files share a common structure. They describe the variables and parameters of the Digivex Motion family of devices.
These files are in ASCII format with fields delimited by " ; ".
Each line is for the definition of a different variable or parameter.

Example (start of DSM.csv file):

no_enreg;nom;index;sous_index;unite_hexa;Min;Max;défaut;type;attribut;observabilité
9;Firmware_Version;10249;0;FFFF;sans;sans;;15;3;17
10;CAN_Address;10250;0;FFFF;0;63;;16;1;17
11;Serial_Number;10251;0;FFFF;sans;sans;;15;1;17
12;Param_12;10252;0;FFFF;0;65535;;12;1;17
13;CAN_Baud_Rate;10253;0;FFFF;0;63;;16;1;17
14;Processor_Type;10254;0;FFFF;sans;sans;C167CR LM;15;3;17

Field description:

no_enreg	gives the recording number to be used in most functions to access the variable or parameter (see PVD 3527, Digivex Motion Directory of Variables, column "no." in the descriptive tables)
nom	gives the symbolic name of the variable or parameter
index	gives the CANopen index number to be used by the software driver for accessing the variable or parameter
sous_index	gives the CANopen sub-index number
unite_hexa	code giving the variable or parameter unit
Min	minimum value of the variable or parameter
Max	maximum value of the variable or parameter
défaut	default value of the variable or parameter
type	code giving the type of the variable or parameter (floating, integer, binary, etc.)
attribut	code indicating whether the variable or parameter can be accessed in read only or in read/write mode
observabilité	code indicating the Parvex tool for observing the variable or parameter

2.5 Memories

The various information communicated to the variable speed drive positioner such as data, parameters or programs, is stored in distinct memory zones.

The **_DM** extension refers to devices in the ***D****IGIVEX* ***M****OTION* family (*DSM*, *DMM*, *DPM*).
The **_PC** extension concerns the PC connected to the *CAN* bus.

2.5.1 EEPROM_DM

A non-volatile storage zone, which is called EEPROM_DM, contains the machine parameters and the user configuration file.
This memory is physically located on the personalization board of the variable speed drive positioner.
A part of the EEPROM_DM is integrated with the motherboard of the variable speed drive positioner. In particular, it memorizes the operating time of the positioner and its serial number.

2.5.2 RAM_DM

When the variable speed drive positioner is powered up, or restored (control restore), the contents of the EEPROM_DM are transferred to a RAM, which is called RAM_DM. This memory is used to exchange data on-line (*ON LINE)* between the positioner and the PC.

2.5.3 FLASH_DM

User programs (assembler programs, source programs, cam tables) are located in a flash_eprom – type storage zone, which is called FLASH_DM. This memory is located on the personalization board and takes up 512Kb of space.

2.5.4 PROG_DM

The executable code contained in the FLASH_DM memory is transferred to a RAM called PROG_DM (216Kb RAM) at power-up of the variable speed drive positioner, or during program application ("Apply programs" command).
User programs found in the PROG_DM memory are executed by the variable speed drive positioner.

2.5.5 FIRM_DM

The operating system software is called *firmware.*
It is located in a flash_eprom –type storage zone, which is called FIRM_DM. This memory is integrated with the motherboard of the variable speed drive positioner.

2.5.6 RAM_PC

When the variable speed drive positioner is connected to the PC, an image of its RAM_DM is transmitted to the RAM in the PC, which is called RAM_PC.

2.5.7 Bulk Storage

Diskettes and hard drives make up the bulk storage unit of the PC. This memory is used to save user data.

unite_hexa :

FFFF	none
FFFE	V
FFFD	A
FFFC	Ω
FFFB	mH
FFFA	Hz
FFF9	s
FFF8	rpm
FFF7	°C
FFF6	marks per rev
FFF5	angular degree
FFF4	%
FFF3	V/1000rpm
FFF2	mA/rpm
FFF1	A/1000rpm
FFF0	kg.m²
FFEF	rpm/V
FFEE	N.m
FFED	N.m/1000rpm
FFEC	%/°C
FFEB	mA
FFEA	angular minute
FFE9	J
FEFF	unit1
FEFE	unit1/s
FEFD	unit1/s²
FEFC	unit1/tr
FEFB	unit1/pulses
FEFA	unit2
FEF9	unit2/s
FEF8	unit2/s²
FEF7	unit2/tr
FEF6	unit2/pulses
FEF5	unit3
FEF4	V/unit3
FEF3	unit4
FEF2	unit4/V
FEF1	pulses
FEF0	1/s
FEEF	ms

type :

0	not defined
1	16-character string
2	not defined
3	64-bit floating IEEE
4	32-bit floating IEEE
5	32-bit signed hexadecimal integer
6	binary (0 or 1) coded on 32-bit hexadecimal integer
7	maximum 16-character string
8	16-character string in the format ± 0.123456789E±01
9	not defined
10	unsigned 32-bit hexadecimal integer
11	signed 32-bit hexadecimal integer
12	unsigned 16-bit hexadecimal integer
13	signed 16-bit hexadecimal integer
14	binary (0 or 1) coded on 16-bit hexadecimal integer
15	maximum 16-character string
16	16-character string in the format ± 0.123456789E±01

attribut :

0	not defined
1	read/write parameter
2	specific parameter
3	read only parameter
4	read only variable
5	read/write variable

Variables and parameters differ only in that:

- variables have initialisation values that the user cannot modify
- parameters have initialisation values that the user can modify

observabilité :

0	cannot be observed
0 / 16 / 17 / 81	cannot be observed with the oscilloscope tool
5 / 13 / 85 / 125 / 253	can be observed with the oscilloscope tool
16 / 17 / 81 / 85 / 125 / 241 / 253	can be observed other than with the oscilloscope tool

3. GUIDE TO FUNCTIONS

3.1 Alphabetical function directory

The following pages document the functions:

- **PxChangeCANBaudRate**
- **PxChangeRS232BaudRate**
- **PxCloseAllStream**
- **PxCloseStream**
- **PxDirectReadParam**
- **PxDirectWriteParam**
- **PxGetFirmLoadProgress**
- **PxOpenStream**
- **PxOpenStreamEx**
- **PxReadOscilloCurve**
- **PxReadParam**
- **PxReadUserProgFile**
- **PxWriteFirmWareFile**
- **PxWriteParam**
- **PxWriteParameterFile**
- **PxWriteUserProgFile**

3.2 Function feedback - error codes

SUCCESS = 0 (no error)

-1	Wrong communication stream number
-3	Communication driver busy. Please begin again later
-4	Communication driver not answering. Please restart application
-5	Driver closed. No communication possible. Restart PxComm.exe
-20	Value beyond limits. Please begin again
-21	Parameter inaccessible
-22	Parameter addressed is read only type
-23	No such parameter
-24	Drive must be at zero speed and torque to write this parameter
-25	File corrupted. Contains syntax error !
-26	Mismatch between value and parameter type. Check value
-27	Check-sum error detected during firmware transfer
-28	Check-sum error. Firmware file corrupted
-30	Time-out error on CANopen bus
-50	Communication problem with CRS232 interface
-51	Communication with CRS232 interface interrupted. Check connection and restart
-100	Communication impossible. Check connections and restart application if necessary
-202	Command being executed. Please try again
-204	Parameter definition file missing or corrupted

3.3 Detailed description of the various functions

PxChangeCANBaudRate

Function

This function requests the driver to change the communication speed the CRS232 interface will use to communicate over the CANopen bus.

Synopsis

DELPHI

function **PxChangeCANBaudRate**(**iStream**: Integer;
BaudRateCode: Integer): Integer; **stdcall**;

C

int **_stdcall PxChangeCANBaudRate**(int **iStream**,
int **BaudRateCode**);

VISUAL BASIC

Private Declare Function **PxChangeCANBaudRate** Lib "pxcomm.dll"
(ByVal **iStream** As Long, ByVal **BaudRateCode** As Long) As Long

Parameters

iStream	Integer type, indicates the identifier of the stream to be used (returned previously by the **PxOpenStream** function)
BaudRateCode	Integer type, contains a code describing the desired communication speed:

0	10 Kbaud	4	250 Kbaud
1	20 Kbaud	5	500 Kbaud
2	50 Kbaud	6	1 Mbaud
3	125 Kbaud		

Feedback

Integer type

< 0	an error has occurred (see error codes § 3.2)
6	operation completed successfully

Comments

The stream identifier to be used must correspond to the CRS232 interface physically connected.

The driver waits for CRS232 clearance before returning to this function.

Attention !

Before using this function, the baud rate on the CANopen network must be changed for each subscriber connected by the following procedure:

- connect just a single device to the network
- change the send / receive speed of this subscriber on the CANopen bus using the PxWriteParam function (write the "Can_Baud_Rate" parameter recording number 13 for a Digivex Motion family positioner drive)
- change the CANopen speed of the CRS232 interface by using the PxChangeCANBaudRate function
- safeguard the subscriber speed change by using the PxWriteParam function (write the "Save_Param" parameter recording number 12295 for a Digivex Motion family positioner drive);
- restore the initial CANopen speed of the CRS232 interface by using the PxChangeCANBaudRate function
- disconnect the device and repeat the procedure for the other devices
- reconnect all the devices
- change the CANopen speed of the CRS232 interface one last time (desired final speed) by using the PxChangeBaudRate function.

PxChangeRS232BaudRate

Function

This function requests the drive to change the communication speed on the serial link between the PC and the CRS232 interface.

Synopsis

DELPHI

function **PxChangeRS232BaudRate**(**iStream**: Integer;
BaudRate: dword): Integer; **stdcall**;

C

int **_stdcall PxChangeRS232BaudRate**(int **iStream**,
DWORD **BaudRate**);

VISUAL BASIC

Private Declare Function **PxChangeRS232BaudRate** Lib "pxcomm.dll"
(ByVal **iStream** As Long, ByVal **BaudRate** As Long) As Long

Parameters

iStream — Integer type, indicates the identifier of the stream to be used (returned previously by the **PxOpenStream** function)

BaudRate — dWord type, indicates the speed to be applied to the serial link between the PC and the CRS232 interface.
Possible baud rate values are:

0	9600 baud	2	57600 baud
1	19200 baud	3	115200 baud

Feedback

Integer type

< 0 — an error has occurred (see error codes § 3.2)

5 — operation completed successfully

Comments

The change request is sent at the old baud rate, with one byte representing the desired baud rate as a parameter. Communication is then interrupted, the link closed and then opened again automatically at the new baud rate.

The driver waits for CRS232 clearance before returning to this function.

PxCloseAllStream

Function

This function requests the CRS232 interface driver to close all communication streams previously opened by calls to the **PxOpenStream** function.

Synopsis

DELPHI

function **PxCloseAllStream**(**dwProcess**: dWord): Integer; **stdcall**;

C

int **_stdcall PxCloseAllStream**(DWORD **dwProcess**);

VISUAL BASIC

Private Declare Function **PxCloseAllStream** Lib "pxcomm.dll"
(ByVal **dwProcess** As Long) As Long

Parameters

dwProcess	dWord type, indicates the caller process identifier (returned previously by the API Win32 **GetCurrentProcessid** function)

Feedback

Integer type

< 0	an error has occurred (see error codes § 3.2)
≥ **0**	operation successfully completed (returns the number of streams closed)

Comments

No further communication is possible on communication streams closed by this function (unless a new stream is opened).

PxCloseStream

Function

This function requests the CRS232 interface driver to close a communication stream previously opened by a call to the **PxOpenStream** function.

Synopsis

DELPHI

function **PxCloseStream**(**dwProcess**: dWord;
iStream: Integer): Integer; **stdcall**;

C

int **_stdcall PxCloseStream**(DWORD **dwProcess,**
int **iStream**);

VISUAL BASIC

Private Declare Function **PxCloseStream** Lib "pxcomm.dll"
(ByVal **dwProcess** As Long, ByVal **iStream** As Long) As Long

Parameters

dwProcess	dWord type, indicates the caller process identifier (returned previously by the API Win32 **GetCurrentProcessid** function)
iStream	Integer type, indicates the identifier of the stream to be closed (returned previously by the **PxOpenStream** function)

Feedback

Integer type

< 0	an error has occurred (see error codes § 3.2)
> 0	operation successfully completed (returns the number of streams closed)

Comments

No further communication is possible on the communication stream closed by this function (unless a new stream is opened).

PxDirectReadParam

Function

This function is exactly the same as **PxReadParam**. The only difference being that the index number, sub-index number and parameter type to be read are specified. This function would be used when the subscriber on which we have just read the parameter is not an µVision terminal or a DIGIVEX Motion variable speed drive positioner (DSM, DMM or DPM).

Synopsis

DELPHI

```
function PxDirectReadParam( iStream: Integer;
                            iNode: Integer;
                            wIndexParam: word;
                            wSubIndexParam: word;
                            iType: Integer;
                            var cArray: array of Char ):Integer; stdcall;
```

C

```
int_stdcall PxDirectReadParam( int iStream,
                               int iNode,
                               WORD wIndexParam,
                               WORD wSubIndexParam,
                               int iType,
                               char cArray[4096],
                               int iSize );
```

VISUAL BASIC

```
Private Declare Function PxDirectReadParam Lib "pxcomm.dll"
    (ByVal iStream As Long, ByVal iNode As Long, ByVal wIndexParam As Long, ByVal
    wSubIndexParam As Long, ByVal iType As Long, ByVal cArray As Long, ByVal iSize As
    Long ) As Long
```

Parameters

iStream	Integer type, indicates the identifier of the stream to be used (previously returned by the **PxOpenStream** function)
iNode	Integer type, indicates the number of the CANopen subscriber concerned by the request
wIndexParam	Word type, indicates the CANopen index number of the parameter to be read
wSubIndexParam	Word type, indicates the CANopen sub-index number of the parameter to be read
iType	integer type, indicates the type of parameter to be read (see § 2.4)
cArray	array of Char type, contains by return, the coded ASCII value of the variable or parameter read
iSize	Size of the cArray (used solely in C and VISUAL BASIC)

Return

Integer type

< 0	an error has occurred (see § 3.2 error codes)
= SUCCESS	operation successfully completed

Comments

see **PxReadParam** function.

PxDirectWriteParam

Function

This function is exactly the same as **PxWriteParam**. The only difference being that the index number, sub-index number and parameter type to be modified are specified. This function would be used when the subscriber on which we have just written the parameter is not an µVision terminal or a DIGIVEX Motion variable speed drive positioner (DSM, DMM or DPM).

Synopsis

DELPHI

```
function PxDirectReadParam( iStream: Integer;
                            iNode: Integer;
                            wIndexParam: word;
                            wSubIndexParam: word;
                            iType: Integer;
                            cArray: array of Char ):Integer; stdcall;
```

C

```
int_stdcall PxDirectReadParam( int iStream,
                               int iNode,
                               WORD wIndexParam,
                               WORD wSubIndexParam,
                               int iType,
                               char cArray[4096],
                               int iSize );
```

VISUAL BASIC

```
Private Declare Function PxDirectReadParam Lib "pxcomm.dll"
    (ByVal iStream As Long, ByVal iNode As Long, ByVal wIndexParam As Long, ByVal
    wSubIndexParam As Long, ByVal iType As Long, ByVal cArray As Long, ByVal iSize As
    Long ) As Long
```

Parameters

iStream	Integer type, indicates the identifier of the stream to be used (previously returned by the **PxOpenStream** function)
iNode	Integer type, indicates the number of the CANopen subscriber concerned by the request
wIndexParam	Word type, indicates the CANopen index number of the parameter to be modified
wSubIndexParam	Word type, indicates the CANopen sub-index number of the parameter to be modified
iType	Integer type, indicates the type of parameter to be modified (see § 2.4)
cArray	array of Char type, contains by return, the coded ASCII value of the variable or parameter to be modified
iSize	Size of the cArray (used solely in C and VISUAL BASIC)

Return

Integer type

< 0	An error has occurred (see § 3.2 error codes)
= SUCCESS	Operation successfully completed

Comments

see **PxWriteParam** function.

PxGetFirmLoadProgress

Function

Further to a request to transfer firmware to a Digivex Motion family positioner drive (**PxWriteFirmWareFile**) this function implements a waiting period and displays the state of progress of the transfer.

Synopsis

DELPHI

function **PxGetFirmLoadProgress**(**iStream**:Integer): Integer; **stdcall**;

C

int **_stdcall PxGetFirmLoadProgress**(int **iStream**);

VISUAL BASIC

Private Declare Function **PxGetFirmLoadProgress** Lib "pxcomm.dll"
(ByVal **iStream** As Long) As Long

Parameters

iStream — Integer type, indicates the identifier of the stream to be used (returned previously by the **PxOpenStream** function)
the stream identifier is the same as that used for starting firmware transfer (**PxWriteFirmWareFile** function)

Feedback

Integer type

The value returned (integer value from 0 to 100) shows what percentage of the Firmware transfer function underway has been completed. The default value (no loading in progress) is 100%.

Comments

The monitoring code using the **PxGetFirmLoadProgress** must be implanted in a different thread to the one that started the firmware transfer function.

PxOpenStream

Function

This function requests the driver to open a communication stream between an application and a CRS232 interface by using the serial port specified in the function call.

Synopsis

DELPHI

```
function PxOpenStream( dwProcess: dWord;
                       pCOM: PChar;
                       BaudRate: dWord;
                       iConfig: Integer ): Integer; stdcall;
```

C

```
int _stdcall PxOpenStream( DWORD dwProcess,
                           LPSTR pCom,
                           DWORD BaudRate,
                           int iConfig );
```

VISUAL BASIC

```
Private Declare Function PxOpenStream Lib "pxcomm.dll"
    (ByVal dwProcess As Long, ByVal pCom As String, ByVal BaudRate as Long, ByVal iConfig
    as Long) As Long
```

Parameters

dwProcess dWord type, indicates the caller process identifier (returned previously by the API Win32 **GetCurrentProcessid** function)

pCOM Pchar type, points on the ASCII character string (with terminal "zero") containing the non-coded name of the serial link port to be used for the stream to be opened (COM1, COM2, etc.)

BaudRate dWord type, indicates the non-coded value of the transmission speed to be used for opening the serial link (9600, 19200, 57600 or 115200)

iConfig Integer type, contains a code identifying the variable and parameter definition file of the subscriber for which the stream will be opened:

0 Param.csv (reserved for internal use)
1 DSM.csv (to be used for addressing a positioner drive of the Digivex Motion family: DSM, DMM or DPM)
2 DTP.csv (to be used for addressing a µVision terminal)

Feedback

Integer type

< 0 an error has occurred (see error codes § 3.2)

= 0 an error has occurred: no more communication streams available

> 0 returns the stream identifier (unique number from 1 to 16)

Comments

This function must be used first before any exchange over the network. It is the opposite of communication stream closure with the **PxCloseStream** function.

If the CRS232 communication interface is not initialised, the driver transfers its firmware, waits for a clearance response, then sends the desired RS232 baud rate. If the specified serial link port is already open (operation effected previously by another stream), the function tests whether the CRS232 interface is properly initialised, initialises it if necessary and returns a unique stream number to the application making the request.

Up to 16 streams can be open at once.

N.B.: initialising the CRS232 interface takes a non-negligible time (a few seconds) but is usually done once only upon energising the unit.

PxOpenStreamEx

Function

This function is exactly the same as **PxOpenStream**. The only difference being that the definition file name of the variables and parameters of the subscriber for which the stream will be opened are specified. This function would be used when the subscriber is not an µVision terminal or a DIGIVEX Motion variable speed drive positioner (DSM, DMM or DPM).

Synopsis

DELPHI

```
function PxOpenStreamEx( dwProcess: dWord;
                         pCOM: PChar;
                         BaudRate: dWord;
                         iConfig: Integer;
                         cFile: array of Char ):Integer; stdcall;
```

C

```
int_stdcall PxOpenStream( DWORD dwProcess,
                          LPSTR pCom,
                          DWORD BaudRate,
                          int iConfig,
                          char cFile[4096],
                          int iSize );
```

VISUAL BASIC

```
Private Declare Function PxOpenStream Lib "pxcomm.dll"
    (ByVal dwProcess As Long, ByVal pCom As String, ByVal BaudRate As Long, ByVal
iConfig
    As Long, ByVal cFile As Long, ByVal iSize As Long ) As Long
```

Parameters

dwProcess	dWord type, indicates the identifier of the calling process (previously returned by the **GetCurrentProcessId** function of the API Win32)
pCom	Pchar type, points to the ASCII character string (at terminal "zero") containing the uncoded name of the serial link port to be used for the stream to be opened (COM1, COM2, etc.)
BaudRate	dWord, indicates the uncoded value of the transmission speed to be used to open the serial link (9600, 19200, 57600 or 115200)
iConfig	Integer type, contains the 0 code
cFile	array of Char type, contains the ASCII character string (at terminal "zero") which indicates the complete definition file name (.csv) of the variables and parameters of the subscriber for which the stream will be opened
iSize	Size of the cFile (used solely in C and VISUAL BASIC)

Return

Integer type

< 0	An error has occurred (see § 3.2 error codes)
= 0	An error has occurred: the communication stream is no longer available
> 0	Returns the stream identifier (one number taken from between 1 and 16)

Comments

see **PxOpenStream** function.

PxReadOscilloCurve

Function

This function sends the CRS232 interface a request to read the oscilloscope curve on the positioner drive corresponding to the specified CANopen subscriber.

Synopsis

DELPHI

```
function PxReadOscilloCurve( iStream: Integer;
                             iNode: Integer;
                             wNumParam: word;
                             wNumSegment: word;
                             var cArray: array of char ): Integer; stdcall;
```

C

```
int _stdcall PxReadOscilloCurve( int iStream,
                                 int iNode,
                                 WORD wNumParam,
                                 WORD wNumSegment,
                                 char cArray[4096],
                                 int iSize);
```

VISUAL BASIC

```
Private Declare Function PxReadOscilloCurve Lib "pxcomm.dll"
    (ByVal iStream As Long, ByVal iNode As Long, ByVal wNumParam As Long, ByVal
    wNumSegment As Long, ByVal cArray As Long, ByVal iSize As Long) As Long
```

Parameters

iStream — Integer type, indicates the identifier of the stream to be used (returned previously by the **PxOpenStream** function)

iNode — Integer type, indicates the number of the CANopen subscriber concerned by the request

wNumParam — Word type, indicates the index number of the variable containing the curb points to be read:

- 1180 for channel 1 (osc_channel0)
- 1181 for channel 2 (osc_channel1)
- 1182 for channel 3 (osc_channel2)
- 1183 for channel 4 (osc_channel3)

wNumSegment — Word type, indicates the number of the curve segment to be read (1 to 128)

cArray — Array of Char type, contains by return, the values of the oscilloscope points read (16 floating values 32 simple precision bits coded in IEEE format) for the segment in question.

iSize — Size of the cArray table (used solely in C and VISUAL BASIC)

Feedback

Integer type

< 0 — an error has occurred (see error codes § 3.2)

= SUCCESS — operation completed successfully

Comments

The driver waits for CRS232 clearance before returning to this function.

Oscilloscope curves are divided into 128 segments of 16 floating values (IEEE format). The "osc_data_ready" variable specifies the number of segments ready for the user to come and read.

The user has access to 10 parameters for defining the oscilloscope function:

Parameter	Recording no.	Description
osc_num0	1184*	integer defining the recording number of the variable associated with channel 1 observable variables are specified in PVD 3527 "Digivex Motion Directory of Variables"
osc_num1	1185*	integer defining the recording number of the variable associated with channel 2 observable variables are specified in PVD 3527 "Digivex Motion Directory of Variables"
osc_num2	1186*	integer defining the recording number of the variable associated with channel 3 observable variables are specified in PVD 3527 "Digivex Motion Directory of Variables"
osc_num3	1187*	integer defining the recording number of the variable associated with channel 4 observable variables are specified in PVD 3527 "Digivex Motion Directory of Variables"
osc_trig_level	1188*	IEEE floating giving the oscilloscope trigger level
osc_trig_channel	1189*	integer defining the channel number used as the trigger (0, 1, 2 or 3)
osc_trig_edge	1190*	binary defining the direction of triggering: 0 positive-going (increasing value) 1 negative-going (decreasing value)
osc_pretrig	1191*	IEEE floating giving the pre-trig value (percentage of trace before trigger) min 0,01 (1%) max 0,99 (99%)
osc_timebase	1192*	Whole number giving the value of the oscilloscope time base (expressed in multiples of 250µs) time base = time period between two samples mini 1 (0.25ms) maxi 16,000 (4s)
osc_trig_on	1193*	binary defining triggering: 0 tripped immediately 1 tripped by trigger

* For a Digivex Motion family positioner drive.

In addition, two variables can start acquisition and indicate that data are ready for reading:

Parameter	Recording no.	Description
osc_on	1194*	1 written to this variable triggers acquisition
osc_data_ready	1195*	integer indicating the number of "segments" ready min 0 max 128

* For a Digivex Motion family positioner drive.

Remark :

The Parvex oscilloscope tool supplied with PME software can also be used. Just pass on the command line the following parameters to the scope.exe executable (the various parameters are separated by a "space" character):

scope.exe DSM 3 COM1 115200 12

Parameter	Example	Description
1	DSM	variable and parameter definition file name without extensions (DSM.csv for a Digivex Motion family positioner drive)
2	3	fixed parameter for internal use
3	COM1	name of the serial link port used (COM1, COM2, etc.)
4	115200	serial link baud rate (9600, 19200, 57600 or 115200)
5	12	CANopen subscriber number of the device in question

PxReadParam

Function

This function sends the CRS232 interface a request to read a variable or parameter on the specified CANopen subscriber.

Synopsis

DELPHI

```
function PxReadParam( iStream: Integer;
                      iNode: Integer;
                      wNumParam: word;
                      var cArray: array of char ): Integer; stdcall;
```

C

```
int _stdcall PxReadParam( int iStream,
                          int iNode,
                          WORD wNumParam,
                          char cArray[4096],
                          int iSize);
```

VISUAL BASIC

```
Private Declare Function PxReadParam Lib "pxcomm.dll"
    (ByVal iStream As Long, ByVal iNode As Long, ByVal wNumParam As Long, ByVal
    cArray As Long, ByVal iSize As Long) As Long
```

Parameters

iStream	Integer type, indicates the identifier of the stream to be used (returned previously by the **PxOpenStream** function)
iNode	Integer type, indicates the number of the CANopen subscriber concerned by the request
wNumParam	Word type, indicates the recording number of the variable or parameter to be read (see PVD 3527, Digivex Motion Directory of Variables, column "no." in the descriptive tables)
cArray	array of Char type, contains when returned the ASCII coded value of the variable or parameter read
iSize	Size of the cArray table (used solely in C and VISUAL BASIC)

Feedback

Integer type

< 0	an error has occurred (see error codes § 3.2)
= SUCCESS	operation completed successfully

Comments

The function translates the value read in the device in an ASCII character string.

Attention: depending on the variable or parameter read, this function can return an ASCII coded value from 2 bytes to 4 Kbytes (usually 16 bytes). The buffer receiving the data must therefore be large enough.

The driver waits for CRS232 clearance before returning to this function.

PxReadUserProgFile

Function

This function sends the CRS232 interface a request to save on the PC the User Program of the positioner drive corresponding to the specified CANopen subscriber.

Synopsis

DELPHI

```
function PxReadUserProgFile( iStream: Integer;
                             iNode: Integer;
                             wNumParam: word;
                             var Filename: PChar ):Integer; stdcall;
```

C

```
int _stdcall PxReadUserProgFile( int iStream,
                                 int iNode,
                                 WORD wNumParam,
                                 LPSTR FileName );
```

VISUAL BASIC

```
Private Declare Function PxReadUserProgFile Lib "pxcomm.dll"
    (ByVal iStream As Long, ByVal iNode As Long, ByVal wNumParam As Long, ByVal
    FileName As String) As Long
```

Parameters

iStream	Integer type, indicates the identifier of the stream to be used (returned previously by the **PxOpenStream** function)
iNode	Integer type, indicates the number of the CANopen subscriber concerned by the request
wNumParam	Word type, indicates the recording number corresponding to a User Program safeguard request ("Data_Prog" number 12288 for a positioner drive)
FileName	Pchar type, points, on return, on an ASCII character string (with terminal "zero") indicating the temporary file name containing the user program read

Feedback

Integer type

< 0	an error has occurred (see error codes § 3.2)
= SUCCESS	operation completed successfully

Comments

The driver creates a temporary file, records the user program downloaded in it, and then returns the name of this file.

The driver waits for CRS232 clearance before returning to this function.

PxWriteFirmWareFile

Function

This function requests the CRS232 interface driver to transfer a Firmware file to a CANopen subscriber (transfer to FIRM_DM memory).

Synopsis

DELPHI

```
function PxWriteFirmWareFile( iStream: Integer;
                              iNode: Integer;
                              wNumParam: Word;
                              FileName: PChar ): Integer; stdcall;
```

C

```
int _stdcall PxWriteFirmWareFile( int iStream,
                                  int iNode,
                                  WORD wNumParam,
                                  LPSTR FileName );
```

VISUAL BASIC

```
Private Declare Function PxWriteFirmWareFile Lib "pxcomm.dll"
    (ByVal iStream As Long, ByVal iNode As Long, ByVal wNumParam As Long, ByVal
    FileName As String) As Long
```

Parameters

iStream	Integer type, indicates the identifier of the stream to be used (returned previously by the **PxOpenStream** function)
iNode	Integer type, indicates the number of the CANopen subscriber concerned by the request
wNumParam	Word type, indicates the recording number corresponding to a Firmware file transfer request ("Data_Firm" number 12291 for a positioner drive)
FileName	Pchar type, points on an ASCII character string (with terminal "zero") indicating the full name of the Firmware file to be transferred

Feedback

Integer type

< 0	an error has occurred (see error codes § 3.2)
= SUCCESS	operation completed successfully

Comments

Attention: This function replaces the "Firmware code" of the device connected with that of the file transmitted. If there is a transmission problem, the old Firmware can be recovered by re-energising the device.

The driver waits for CRS232 clearance before returning to this function.

PxWriteParam

Function

This function sends the CRS232 interface a request to write a variable or parameter to the specified CANopen subscriber (transfer to RAM_DM memory)

Synopsis

DELPHI

```
function PxWriteParam( iStream: Integer;
                       iNode: Integer;
                       wNumParam: word;
                       var cArray: array of char ): Integer; stdcall;
```

C

```
int _stdcall PxWriteParam( int iStream,
                           int iNode,
                           WORD wNumParam,
                           char cArray[4096],
                           int iSize);
```

VISUAL BASIC

```
Private Declare Function PxWriteParam Lib "pxcomm.dll"
    (ByVal iStream As Long, ByVal iNode As Long, ByVal wNumParam As Long, ByVal cArray
    As Long, ByVal iSize As Long) As Long
```

Parameters

iStream	Integer type, indicates the identifier of the stream to be used (returned previously by the **PxOpenStream** function)
iNode	Integer type, indicates the number of the CANopen subscriber concerned by the request
wNumParam	Word type, indicates the recording number corresponding to the variable or parameter to be modified (see PVD 3527 Digivex Motion Directory of Variables, column "no." in the descriptive tables)
cArray	Array of Char type, contains the coded ASCII value of the variable or the parameter to be modified
iSize	Size of the cArray table (used solely in C and VISUAL BASIC)

Feedback

Integer type

< 0	an error has occurred (see error codes § 3.2)
= SUCCESS	operation completed successfully

Comments

The value sent to the function must be a character string (e.g. "+1,444" or "AXIS1"). If this value does not match the expected format for the variable or parameter to be modified, an error is returned by the function.

The driver waits for CRS232 clearance before returning to this function.

Remark: Use the PxWriteParam function to write the '1' text in the "Save_Param" parameter (DIGIVEX MOTION variable speed drive positioner registration number 12295) and save previously changed parameter(s) in EEPROM_DM memory. This instruction backs-up all RAM_DM memory parameters in EEPROM_DM memory.

PxWriteParameterFile

Function

This function requests the CRS232 interface drive to transfer a Parameter file to a CANopen subscriber (transfer to RAM_DM memory).

Synopsis

DELPHI

function **PxWriteParameterFile**(**iStream**: Integer;
iNode: Integer;
FileName: PChar): Integer; **stdcall**;

C

int **_stdcall PxWriteParameterFile**(int **iStream**,
int **iNode**,
LPSTR **FileName**);

VISUAL BASIC

Private Declare Function **PxWriteParameterFile** Lib "pxcomm.dll"
(ByVal **iStream** As Long, ByVal **iNode** As Long, ByVal **FileName** As String) As Long

Parameters

IStream	Integer type, indicates the identifier of the stream to be used (returned previously by the **PxOpenStream** function)
INode	Integer type, indicates the number of the CANopen subscriber concerned by the request
FileName	Pchar type, points on an ASCII character string (with terminal "zero") indicating the full name of the Parameter file to be transferred

Feedback

Integer type

< 0	an error has occurred (see error codes § 3.2)
= SUCCESS	operation completed successfully

Comments

The parameter file to be written must have the following syntax:

<Parameters>
N°_du_paramètre_en_décimal=*Valeur_du_paramètre_en_ASCII* ;*Nom_du_paramètre_en_ASCII*
</Parameters>

Example :

<Parameters>
51=LX310BS;Nmot
52=DIGIVEX 4/8;Nvar
53=1;TypC
54=1;TypF
55=280;U
</Parameters>

The driver waits for CRS232 clearance before returning to this function.

Remark: Use the PxWriteParam function to write the '1' text in the "Save_Param" parameter (DIGIVEX MOTION variable speed drive positioner registration number 12295) and save all parameters present in RAM_DM memory in EEPROM_DM memory.

PxWriteUserProgFile

Function

This function requests the CRS232 interface drive to transfer a User Program file to a positioner drive corresponding to the specified CANopen subscriber (transfer to FLASH_DM memory).

Synopsis

DELPHI

```
function PxWriteUserProgFile( iStream: Integer;
                              iNode: Integer;
                              wNumParam: Word;
                              FileName: PChar ): Integer; stdcall;
```

C

```
int _stdcall PxWriteUserProgFile( int iStream,
                                  int iNode,
                                  WORD wNumParam,
                                  LPSTR FileName );
```

VISUAL BASIC

```
Private Declare Function PxWriteUserProgFile Lib "pxcomm.dll"
    (ByVal iStream As Long, ByVal iNode As Long, ByVal wNumParam As Long, ByVal
    FileName As String) As Long
```

Parameters

iStream	Integer type, indicates the identifier of the stream to be used (returned previously by the **PxOpenStream** function)
iNode	Integer type, indicates the number of the CANopen subscriber concerned by the request
wNumParam	Word type, indicates the recording number corresponding to a User Program transfer request ("Data_Prog" number 12288 for a positioner drive)
FileName	Pchar type, points on an ASCII character string (with terminal "zero") indicating the full name of the User Program file to be transferred

Feedback

Integer type

< 0	an error has occurred (see error codes § 3.2)
= SUCCESS	operation completed successfully

Comments

This file is a special format and must come from the Parvex program editor only.
The driver waits for CRS232 clearance before returning to this function.

Remarks :

- Before transferring a new User Program file, the user program (FLASH_DM) memory of the positioner drive must first be cleared: use the PxWriteParam function to write the 'DELE' text in the "Erase_Prog" parameter (recording number 12290 for a DIGIVEX Motion family positioner drive).
- "To apply" the User Program : use the PxWriteParam function to write the 'PROG' text in the "Cmd_Prog" parameter (recording number 12289 for a DIGIVEX Motion family positioner drive). This instruction transfers FLASH_DM memory executable code to PROG_DM memory.

4. PROGRAMMING EXAMPLES

Various examples of the uses of D.L.L. features are supplied with **PME Tool Kit**. They can be found on the PME installation CD-ROM in the tool_kit directory.

These examples are presented solely for teaching purposes. Parvex cannot under any circumstances be held responsible for any problems relating to any use made of them.